HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 10012354 -1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Calvin Selig et al

Confirmation No.: 6534

Application No.: 09/823505

Examiner: quillen, Allen E

Filing Date: Mar 31, 2001

Group Art Unit: 2676

Title: Technique For Eliminating Stale Information From A Computer Graphics Buffer

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Sir:

Transmitted herewith in **triplicate** is the Appeal Brief in this application with respect to the Notice of Appeal filed on **03/19/2004**.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) $330.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

( ) (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

( )	one month	$110.00
( )	two months	$420.00
( )	three months	$950.00
( )	four months	$1480.00

( ) The extension fee has already been filled in this application.

(X) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account **08-2025** the sum of **$330.00**. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

(X) I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450. Date of Deposit: **05/19/2004**

OR

( ) I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number _____ on _____

Number of pages: 76 pages including this sheet.

Typed Name: Kevin M. Hart

Signature: Kevin M. Hart

Respectfully submitted,

Calvin Selig et al

By Kevin M. Hart

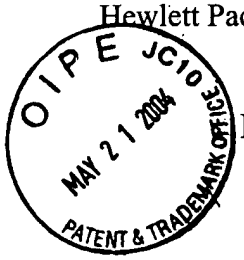
Kevin Hart

Attorney/Agent for Applicant(s)
Reg. No. **36,823**

Date: **05/19/2004**

Telephone No.: **970 898 7057**

Hewlett Packard Docket No.: 10012354-1 PATENT



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
Board of Patent Appeals and Interferences

In re Application of: **Calvin Selig, et al.**	Art Unit: **2676**
Serial No.: **09/823,505**	Examiner: **Quillen, Allen E.**
Filed: **March 31, 2001**	
For: **Technique for Eliminating Stale Information from a Computer Graphics Buffer**	

RECEIVED
MAY 2 6 2004
Technology Center 2600

APPEAL BRIEF

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Applicant (hereinafter "Appellant") appeals from examiner Quillen's final rejection mailed 11/24/2003 (hereinafter "the final rejection"). A Notice of Appeal was filed by facsimile on 3/19/2004 with a petition and fee for a one-month extension of time. Appellant believes no further extensions of time are required; but if further extensions are required, the Commissioner is hereby petitioned to grant such extensions and to charge the requisite fees to Appellant's deposit account no. 08-2025.

Real party in interest

The real party in interest in this appeal is the assignee of the subject patent application: Hewlett-Packard Development Company, a Texas limited partnership having its principal place of business in Houston.

Related appeals and interferences

Also under appeal is Appellant's patent application serial number 09/823,660, filed March 31, 2001, titled "Fast Clear Technique for Regions Having Subregions" (hereinafter "the '660 application"). The '660 application and the present application are related in the sense that they disclose related subject matter. They are not related as parent/child, divisionals, continuations or continuations-in-part.

Status of claims

Claims 1-43 are pending as they were originally presented. They have been finally rejected and are the subject of this appeal.

Status of amendments

No amendments were submitted after the final rejection.

Concise summary of invention

"Fast clear" is a conventional computer graphics technique wherein a value can effectively be set for all of the pixels in a display region without having to write the value to each of the memory locations in a graphics buffer that correspond to the region. Although the conventional fast clear technique provides a performance advantage relative to "brute-force" clearing, it also imposes the limitation that the same clear technique must be employed throughout the life span of a display region in which it is used. This means that, for regions

in which conventional fast clear is employed, operations cannot be performed if they would require the graphics buffer contents to match the image being displayed. And for non fast clear regions, the performance benefits provided by the fast clear technique are not realized. In this context, Appellant has improved on the conventional fast clear technique: Appellant has discovered that it is possible to switch from a fast clear mode to a non fast clear mode during the life span of a display region provided that stale information is first eliminated from the display region prior to the switch. Therefore, Appellant claims a novel and efficient method for eliminating stale information from a computer graphics buffer.

In one embodiment, the method includes (1) reading a clear count associated with a pixel in the graphics buffer; (2) comparing the pixel's clear count with a current clear count; and (3) if the pixel's clear count is not current, writing a predetermined value to the pixel.

In another embodiment, the method includes performing a block transfer operation on pixels in the graphics buffer wherein the source region and the destination region for the block transfer operation are actually the same region. For each pixel in the region, the block transfer operation performs the following steps: (1) reading a clear count associated with the pixel; (2) comparing the pixel's clear count with a current clear count; and (3) if the pixel's clear count is not current, writing a predetermined value to the pixel.

In another embodiment, the method includes: (1) rendering in a region of a graphics buffer using a fast clear mode; (2) determining that the fast clear mode should be discontinued; and (3) eliminating stale information from the region by (for each pixel in the region) reading a clear count associated with the pixel, comparing the pixel's clear count with a current clear count, and if the pixel clear count is not current, writing a predetermined value to the pixel.

Issues

The issue on appeal is whether claims 1-43 are anticipated under 35 U.S.C. § 102(e)

by U.S. patent 6,348,919 ("Murphy").

Grouping of claims

Claims 1-7, 11, 27-29 and 32 will stand or fall together.

Claims 12, 14-16, 33 and 35-37 will stand or fall together.

Claims 17-18, 20-22, 26, 38-39 and 43 will stand or fall together.

All other claims will be argued separately.

Argument

THE EXAMINER'S FINDING OF ANTICIPATION IS ERRONEOUS BECAUSE AT LEAST ONE ELEMENT IN EACH OF CLAIMS 1-43 IS ABSENT FROM THE MURPHY REFERENCE.

1. The Murphy Reference

In the context of fast clear, the Murphy reference discloses nothing more than the well-known "striping" technique that is described in Applicant's own specification and acknowledged therein as conventional:

> Any conventional fast clear technique, such as a striping technique, may be utilized to handle clear commands for region 800. In accordance with a striping technique, region 800 may be divided into stripes 802. Alternatively, region 800 may be divided into vertical columns, or region 800 may be further divided into a matrix of rectangles. (The terms "subdivisions" and "stripes" and "striping" are used interchangeably herein to refer to any and all of these alternative stripes, columns or rectangles.) Responsive to each clear command, the current clear count for region

> 800 may be incremented, and then an actual clear may be performed in one of stripes 802. The one stripe 802 chosen for actual clearing may change according to a cyclic schedule so that all of stripes 802 will have been actually cleared at the completion of the cycle. The benefit of the striping technique is that an actual full clear of the entire striped area need not be performed in response to any one clear command, provided the number of stripes in the region is not greater than the maximum value of the current clear count register for the region; instead, the full clear is amortized over several clear commands.

Application at page 12, lines 8-22. Compare the above with Murphy's disclosure:

> The fast clear mechanism provides a method where the cost of clearing the depth and stencil buffers can be amortized over a number of clear operations issued by the application. This works as follows:
>
> The window is divided up into n regions, where n is the range of the frame counter (16 or 256). Every time the application issues a clear command the reference frame counter is incremented ... and the nth region is cleared only. The clear updates the depth and/or stencil buffers to the new values and the frame count buffer with the reference value. This region is much smaller than the full window and hence takes less time to clear.

Murphy at col. 26, lines 6-19.

Applicant does not purport herein to have invented striping. While Applicant's invention may be employed in conjunction with striping, striping does not anticipate Applicant's invention as claimed.

2. The Standard for Anticipation

Anticipation requires that each and every element of the claimed invention be disclosed in a single prior art reference. In re Paulsen, 31 U.S.P.Q.2d 1671 (Fed. Cir. 1994); In re Spada, 15 U.S.P.Q.2d 1913 (Fed. Cir. 1990); In re Donohue, 226 U.S.P.Q. 619 (Fed. Cir. 1985). The corollary of this rule is that absence from the reference of any claimed element negates anticipation. Kloster Speedsteel AB v. Crucible Inc., 230 U.S.P.Q. 81 (Fed. Cir. 1986).

Not only must each of the claim elements be disclosed in the single prior art reference, they must also be arranged in the reference as they are arranged in the claim. Richardson v. Suzuki Motor Co., 9 U.S.P.Q.2d 1913 (Fed. Cir. 1989); Carella v. Starlight Archery & Pro Line Co., 231 U.S.P.Q. 644 (Fed. Cir. 1986); Lindemann Maschinenfabrik v. American Hoist & Derrick Co., 221 U.S.P.Q. 481 (Fed. Cir. 1984); Connell v. Sears, Roebuck & Co., 220 U.S.P.Q. 193 (Fed. Cir. 1983). In short, anticipation requires identity between what is claimed and what is disclosed in the reference. Tyler Refrigeration v. Kysor Indus. Corp., 227 U.S.P.Q. 845 (Fed. Cir. 1987); Shatterproof Glass Corp. v. Libbey-Owens Ford Co., 225 U.S.P.Q. 634 (Fed. Cir. 1985); Rosemount, Inc. v. Beckman Instruments, Inc., 221 U.S.P.Q. 1 (Fed. Cir. 1984); Kalman v. Kimberly-Clark Corp., 218 U.S.P.Q. 781 (Fed. Cir. 1983).

3. Claims 1-7, 11, 27-29 and 32

MURPHY DOES NOT DISCLOSE WRITING A PREDETERMINED VALUE TO THE PIXEL LOCATION IN THE BUFFER IF THE CLEAR COUNT VALUE OF THE PIXEL DOES NOT EQUAL THE CURRENT CLEAR COUNT.

Claims 1 and 27 require *reading a clear count value associated with a pixel location in a graphics buffer, comparing the clear count value with a current clear count, and if the clear count value does not equal the current clear count, writing a predetermined value to*

the pixel location.

Because Murphy discloses only the conventional use of fast clear as explained above, his disclosure does not include ***writing*** a predetermined value ***to the pixel location in the buffer*** if the clear count value of the pixel does not equal the current clear count. Only such a step could have the possibility of eliminating stale information from the graphics buffer as required by Appellant's claims. Indeed, Murphy never discloses writing any values to pixel locations in the buffer in response to a comparison of the pixel's clear count with a current clear count. Instead, Murphy teaches only the conventional use of fast clear, which is to return one of two values in response to a ***read*** from the graphics buffer: If the pixel's clear count is current, then the pixel's actual value is returned in response to the read request. But if the pixel's value is not current based on a comparison with the current clear count, then a predetermined value is substituted for–is returned instead of--the pixel's actual stale value:

> When the localbuffer is subsequently read and the frame count is found to be the same as the reference frame count ... the localbuffer data is used directly. However, if the frame count is found to be different from the reference frame count ... the data which would have been written, if the localbuffer had been cleared properly, is ***substituted for the stale data returned from the read.*** Any new writes to the localbuffer will set the frame count to the reference value so the next read on this pixel works normally without the substitution. ...
>
> When clear data is ***substituted for real memory data*** (during normal rendering operations) the depth write mask and stencil write masks are ignored to mimic the OpenGL operation when a buffer is cleared.

Murphy at col. 26, lines 20-27 (emphasis added). Murphy's substitution happens only in the read path, which is conventional in fast clear systems. Murphy does not disclose *writing the predetermined value back to the pixel location in the buffer.* To do so would destroy the very

performance advantage of the fast clear technique Murphy is advocating.

The examiner also cited Murphy at col. 5, line 46 to col. 6, line 9 as a basis for rejection. But this latter excerpt is inapposite to the subject matter of Appellant's claims. The Murphy reference is directed to a graphics rendering system that uses a single unified physical memory space to store both depth values and color values. See, e.g., Murphy at abstract and at col. 5, lines 27-37. The excerpt from col.'s 5-6 relates neither to clearing in general, nor to fast clear in particular. Rather, the excerpt simply addresses the problem of synchronizing reads and writes to Murphy's unified memory space when rendering graphics primitives. While the excerpt makes use of the word "stale," it is clear from the context that Murphy is not referring to pixels whose clear count does not equal a current clear count. Instead, Murphy's use of the term stale refers simply to the problem of synchronizing reads and writes during the rendering of graphics primitives. Specifically, when a write to an address is issued before a read to the same address, it is common to say the data would be "stale" if it is read before the write command is executed:

> If random reads and writes were done there would be a danger of stale data in the read unit being used after a location had been updated by the write unit. ...
>
> It is possible that consecutive primitives could touch the same pixel, so safeguards are needed to make sure that all writes for one primitive have completed before reads begin for the next.

Murphy at col. 59, lines 48-57. It is plain to see that this portion of Murphy does not disclose writing a value to a pixel location *responsive to a comparison of the pixel's clear count value with a current clear count value.*

Therefore, the examiner has not established that Murphy discloses ***writing*** a predetermined value ***to the pixel location in the buffer*** if the clear count value of the pixel

does not equal a current clear count, as is required by Appellant's claims.

It is true that a brute-force clear operation such as the "per-pixel write" described in Murphy at col. 26, lines 40-42, does write a predetermined value to pixel locations in the buffer. But such a brute-force per-pixel write involves no comparison between the pixel clear count values and the current clear count, and Murphy does not suggest that it does. Consequently, the rejection of claims 1-7, 11, 27-29 and 32 should be withdrawn.

4. Claims 12, 14-16, 33 and 35-37

MURPHY DOES NOT DISCLOSE PERFORMING THE READING, COMPARING AND CONDITIONAL WRITING STEPS AS PART OF A BLOCK TRANSFER OPERATION ON PIXEL LOCATIONS IN THE BUFFER WHEREIN THE SOURCE AND DESTINATION REGIONS FOR THE BLOCK TRANSFER OPERATION ARE THE SAME.

Claims 12 and 33 are not anticipated by Murphy for all of the reasons stated above in relation to claims 1 and 27. In addition, claims 12 and 33 introduce the further limitation that the reading, comparing and conditional writing steps are performed as part of *a block transfer operation on pixel locations in the buffer wherein a source region and a destination region for the block transfer operation are the same.*

In the embodiment of claims 12 and 33, pixel locations in the region are read, their clear count values compared with a current clear count value, and then conditionally a value is written back to the same pixel locations read--all in the context of a block transfer operation wherein data are not transferred to a different location, but instead are conditionally written back to their own original location. Such a novel technique yields high performance because it utilizes in a novel way certain optimizations that are generally embodied in block

transfer hardware and/or software.

Murphy does not anticipate Appellant's technique because (1) as Appellant established above, Murphy does not disclose writing a predetermined value to the pixel location in the buffer responsive to a comparison of the pixel's clear count value with a current clear count, and (2) Murphy's discussion of block transfer operations merely encompasses the conventional use of block transfer hardware–namely, to copy bits from a non-visible to a visible area of memory, with no consideration given to the value of a clear count for any pixel represented by the bits.

In an effort to supply Appellant's claimed novel use of block transfer hardware, the examiner cited Murphy as follows:

> For arbitrary screen sizes, for instance when rendering to '*offscreen*' memory such as bitmaps the next largest width from the table must be chosen. The difference between the table width and the bitmap width will be an unused strip of pixels down the right hand side of the bitmap.
>
> Note that such bitmaps can be *copied to the screen* only as a series of scanlines rather than as a rectangular block. However, often windowing systems store *offscreen* bitmaps in rectangular regions which use the same stride as the screen. In this case, normal bitblts can be used.

Murphy at col. 28, line 66 to col. 29, line 3, and at col. 27, lines 30-39 (emphasis added). The cited excerpts from Murphy do not mention copying from a source to a destination region wherein the source and destination are in fact the same region. Rather, the excerpts contemplate copying bits from a source to a destination region wherein the source and destination regions are different (i.e., from a non-visible to a visible region). Nor do the excerpts make any mention of comparing pixel clear count values with a current clear count value during the block transfer. But these limitations are required by claims 12 and 33.

Therefore, the examiner has not established anticipation of claims 12 and 33. Consequently, the rejection of claims 12, 14-16, 33 and 35-37 should be withdrawn.

5. Claims 17-18, 20-22, 26, 38-39 and 43

MURPHY DOES NOT DISCLOSE PERFORMING THE READING, COMPARING AND CONDITIONAL WRITING STEPS AFTER RENDERING IN A REGION OF INTEREST USING A FAST CLEAR MODE AND THEN DETERMINING THAT THE FAST CLEAR MODE SHOULD BE DISCONTINUED.

Claims 17 and 38 are not anticipated by Murphy for all of the reasons stated above in relation to claims 1 and 27. In addition, claims 17 and 38 introduce the further limitation that the reading, comparing and conditional writing steps occur *after rendering in a region of interest using a fast clear mode and then determining that the fast clear mode should be discontinued.*

In the embodiment of claims 17 and 38, Appellant has specified a particular time when the elimination of stale information from the graphics buffer should take place–namely, when it becomes desirable to switch from a fast clear mode to a non fast clear mode during the life span of the region.

It was established above that Murphy does not disclose Appellant's reading, comparing and conditional writing steps for eliminating stale information from a graphics buffer. It is also true that Murphy does not disclose that these steps should be performed at any particular *time*. In an attempt to supply the requisite timing, the examiner cited Murphy as follows:

The situation where the stencil planes only are 'cleared' using the fast clear method,

> then some rendering is done and then the depth planes are 'cleared' using the fast clear will leave ambiguous pixels in the localbuffer. The driver software will need to catch this situation, and fall back to using a per pixel write to do the second clear.

Murphy at col. 26, lines 33-42. In this excerpt (which excerpt was also discussed above in relation to claims 1 and 12), Murphy identifies a scenario in which the fast clear technique would not be appropriate. The scenario is when the depth planes must be cleared. But he does not teach eliminating stale information from the depth planes according to Appellant's method. Appellant's method would require reading, comparing and conditionally writing only to those pixels whose clear count values are not current. Instead, Murphy teaches handling the scenario by performing a "brute-force" clear in which a value is written into every pixel of the depth plane, specifically without regard for any clear count values that might be associated with those pixels. This teaching is in direct conflict with Appellant's claimed invention, which requires comparing the clear count values of the pixels with a current clear count prior to conditionally writing new information into the pixels.

Therefore, Murphy does not anticipate Appellant's claims 17 and 38. Consequently, the rejection of claims 17-18, 20-22, 26, 38-39 and 43 should be withdrawn.

6. Other Claims

Claims 8, 23, 30 and 41 add the further limitation that the reading, comparing and conditional writing steps are performed using a block transfer operation wherein the source and destination regions of the block transfer operation are the same. Appellant established above in section 4 that Murphy does not disclose this further limitation. Therefore, Murphy does not anticipate either claims 8, 23, 30 or 41. The rejection of these claims should be withdrawn.

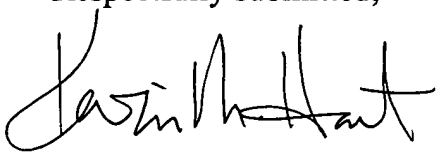
Claims 9-10, 13, 24-25, 31, 34 and 42 add the further limitation that a value is written to each pixel during elimination of stale information from the region; but the value written depends on the outcome of the comparison step. If the comparison indicates that a pixel's clear count is not current, then a predetermined value is written into the pixel; but if the comparison indicates that the pixel's clear count is current, then the pixel's *own previous value is written back into the pixel.* Appellant established above that Murphy does not disclose Appellant's reading, comparing and conditional writing method in any respect. Therefore, Murphy certainly does not disclose writing a pixel's own previous value back into the pixel responsive to the comparison step. Consequently, Murphy does not anticipate any of claims 10 9-10, 13, 24-25, 31, 34 and 42. The rejection of those claims should be withdrawn.

Claims 19 and 40 add the further limitation of, after eliminating stale information from the region, *determining that fast clear mode may be resumed, and resuming operation of fast clear mode.* Appellant established above that Murphy does not disclose Appellant's method of eliminating stale information from a region at any time, let alone after determining that fast clear mode should be discontinued. In addition, Murphy does not disclose resuming fast clear mode after eliminating the stale information in the requisite way. The examiner cited Murphy at col. 26, lines 34-47 in an effort to supply this limitation, but Appellant established above that the cited lines from Murphy contemplate only a "brute-force" clear of the depth plane. They do not contemplate an exit from and a resumption to the fast clear mode, and they do not contemplate a reading/comparing/conditionally writing elimination procedure as is required by Appellant's claims. Consequently, Murphy does not anticipate either claim 19 or claim 40, and the rejection of those claims should be withdrawn.

Conclusion

In view of the above, Appellant requests reversal of the examiner's 11/24/2003 decision finally rejecting claims 1-43.

Respectfully submitted,



Kevin M. Hart
Attorney for Appellant
Reg. No. 36,823
(970)898-7057

Date: 5/19/2004

APPENDIX

Claims on Appeal

1. A method of eliminating stale information from a computer graphics buffer, comprising:

reading a clear count value associated with a pixel location in the buffer;

comparing the clear count value with a current clear count; and

if the clear count value does not equal the current clear count, writing a predetermined value to the pixel location in the buffer.

2. The method of claim 1, wherein:

the clear count value is read from the pixel location in the buffer.

3. The method of claim 1, wherein:

the predetermined value is a color value.

4. The method of claim 1, wherein:

the predetermined value is a z value.

5. The method of claim 1, wherein:

the predetermined value and the current clear count are stored in storage structures of a fast clear system.

6. The method of claim 1, wherein:

the steps are performed for each of the pixels defining a region of interest in the buffer.

7. The method of claim 6, wherein:

the color value is the same for all of the pixels defining the region of interest.

8. The method of claim 6, wherein:

the steps are performed using a block transfer operation wherein a source region and a destination region both correspond to the region of interest.

9. The method of claim 8, wherein:

all of the pixels in the region of interest are read and written during the block transfer operation; and

for a given pixel, if the clear count value equals the current clear count, a stored value read from the pixel location is written back to the pixel location.

10. The method of claim 1, further comprising:

reading a stored value from the pixel location; and

if the clear count value equals the current clear count, writing the stored value back to the pixel location.

11. The method of claim 1, wherein:

the writing step comprises replacing the clear count value with the current clear count.

12. A method of eliminating stale information from a computer graphics buffer, comprising:

performing a block transfer operation on pixel locations of the buffer;

wherein a source region and a destination region for the block transfer operation are the same; and

wherein, for each pixel location, the block transfer operation comprises:

reading a clear count value associated with the pixel location;

comparing the clear count value with a current clear count; and

if the clear count value does not equal the current clear count, writing a predetermined value to the pixel location.

13. The method of claim 12, further comprising:

if the clear count value equals the current clear count, writing a stored value read from the pixel location back to the pixel location.

14. The method of claim 12, wherein:

the clear count value is read from the pixel location in the buffer.

15. The method of claim 12, wherein:

the predetermined value and the current clear count are stored in storage structures of a fast clear system.

16. The method of claim 12, wherein:

the writing step comprises replacing the clear count value with the current clear count.

17. A method of eliminating stale information from a buffer of a computer graphics system, comprising:

using a fast clear mode, rendering in a region of interest within the buffer;

determining, responsive to a state of the computer graphics system, that the fast clear mode should be discontinued; and

for each pixel location in the region of interest:

reading a clear count value associated with the pixel location;

comparing the clear count value with a current clear count; and

if the clear count value does not equal the current clear count, writing a predetermined value to the pixel location.

18. The method of claim 17, wherein the region of interest is a window.

19. The method of claim 17, further comprising:

determining, responsive to a state of the computer graphics system, that the fast clear mode may be resumed; and

resuming operation in the fast clear mode.

20. The method of claim 17, wherein:

the clear count value is read from the pixel location in the buffer.

21. The method of claim 17, wherein:

the predetermined value represents a background color.

22. The method of claim 17, wherein:

the predetermined value and the current clear count are stored in storage structures of a fast clear system.

23. The method of claim 17, wherein:

the reading and writing steps are performed using a block transfer operation wherein a source region and a destination region of the block transfer operation both correspond to the region of interest.

24. The method of claim 23, wherein:

all of the pixels in the region of interest are read and written during the block transfer operation; and

for a given pixel, if the clear count value equals the current clear count, a stored value read from the pixel location is written back to the pixel location.

25. The method of claim 17, further comprising:

reading a stored value from the pixel location; and

if the clear count value equals the current clear count, writing the stored value back to the pixel location.

26. The method of claim 17, wherein:

the writing step comprises replacing the clear count value with the current clear count.

27. Computer program code embodied in a machine-readable storage or transmission medium which, when executed on a computer, causes the computer to perform a method of eliminating stale information from a computer graphics buffer, comprising:

reading a clear count value associated with a pixel location in the buffer;

comparing the clear count value with a current clear count; and

if the clear count value does not equal the current clear count, writing a predetermined value to the pixel location in the buffer.

28. The computer program code of claim 27, wherein:

the steps are performed for each of the pixels defining a region of interest in the buffer.

29. The computer program code of claim 28, wherein:

the predetermined value is the same for all of the pixels defining the region of interest.

30. The computer program code of claim 28, wherein:

the steps are performed using a block transfer operation wherein a source region and a destination region both correspond to the region of interest.

31. The computer program code of claim 30, wherein:

all of the pixels in the region of interest are read and written during the block transfer operation; and

for a given pixel, if the clear count value equals the current clear count, a stored value read from the pixel location is written back to the pixel location.

32. The computer program code of claim 27, wherein:

the writing step comprises replacing the clear count value with the current clear count.

33. Computer program code embodied in a machine-readable storage or transmission medium which, when executed on a computer, causes the computer to perform a method of eliminating stale information from a computer graphics buffer, comprising:

performing a block transfer operation on pixel locations of the buffer;

wherein a source region and a destination region for the block transfer operation are the same; and

wherein, for each pixel location, the block transfer operation comprises:

reading a clear count value associated with the pixel location;

comparing the clear count value with a current clear count; and

if the clear count value does not equal the current clear count, writing a predetermined value to the pixel location.

34. The computer program code of claim 33, further comprising:

if the clear count value equals the current clear count, writing a stored value read from the pixel location back to the pixel location.

35. The computer program code of claim 33, wherein:

the clear count value is read from the pixel location in the buffer.

36. The computer program code of claim 33, wherein:

the predetermined value and the current clear count are stored in storage structures of a fast clear system.

37. The computer program code of claim 33, wherein:

the writing step comprises replacing the clear count value with the current clear count.

38. Computer program code embodied in a machine-readable storage or transmission medium which, when executed on a computer, causes the computer to perform a method of eliminating stale information from a buffer of a computer graphics system, comprising:

using a fast clear mode, rendering an image in a region of interest within the buffer;

determining, responsive to a state of the computer graphics system, that the fast clear mode should be discontinued; and

for each pixel location in the region of interest:

reading a clear count value associated with the pixel location;

comparing the clear count value with a current clear count; and

if the clear count value does not equal the current clear count, writing a predetermined value to the pixel location.

39. The computer program code of claim 38, wherein the region of interest is a window.

40. The computer program code of claim 38, further comprising:

determining, responsive to a state of the computer graphics system, that the fast clear mode may be resumed; and

resuming operation in the fast clear mode.

41. The computer program code of claim 38, wherein:

the reading and writing steps are performed using a block transfer operation wherein a source region and a destination region of the block transfer operation both correspond to the region of interest.

42. The computer program code of claim 41, wherein:

all of the pixels in the region of interest are read and written during the block transfer operation; and

for a given pixel, if the clear count value equals the current clear count, a stored value read from the pixel location is written back to the pixel location.

43. The computer program code of claim 38, wherein:

the writing step comprises replacing the clear count value with the current clear count.